

České vysoké učení technické v Praze  
Fakulta stavební  
Katedra geomatiky



Cvičení 1

## Úloha č. 1: Geometrické vyhledávání bodu

*Bc. Petr Poskočil a Bc. Marek Fáber*

Vyučující: doc. Ing. Tomáš Bayer, Ph.D.

Studijní program: Geodézie a kartografie, Navazující magisterský

Obor: Geomatika

15. října 2019

# Obsah

<b>1</b>	<b>Zadání</b>	<b>1</b>
1.1	Údaje o bonusových úlohách . . . . .	2
<b>2</b>	<b>Popis problému</b>	<b>3</b>
<b>3</b>	<b>Popis použitých algoritmů</b>	<b>4</b>
3.1	Winding Algorithm . . . . .	4
3.1.1	Problematické situace u Winding Algorithm . . . . .	5
3.1.2	Implementace algoritmu . . . . .	5
3.2	Ray Crossing Algorithm . . . . .	6
3.2.1	Problematické situace u Ray Crossing Algorithm . . . . .	6
3.2.2	Implementace metody . . . . .	7
<b>4</b>	<b>Vstup a výstup aplikace</b>	<b>8</b>
4.1	Vstup . . . . .	8
4.2	Výstup . . . . .	8
4.3	Prostředí . . . . .	9
<b>5</b>	<b>Dokumentace tříd a metod</b>	<b>13</b>
5.1	Algorithms . . . . .	13
5.2	Draw . . . . .	13
5.3	Widget . . . . .	14
<b>6</b>	<b>Závěr</b>	<b>15</b>
6.1	Neřešené problémy a náměty . . . . .	15
	<b>Literatura</b>	<b>16</b>
<b>A</b>	<b>Zdrojový kód aplikace</b>	<b>17</b>
<b>B</b>	<b>Vstupní a výstupní data</b>	<b>18</b>
<b>C</b>	<b>Aplikace - binární soubor</b>	<b>19</b>

# Kapitola 1

## Zadání

### Úloha č. 1: Geometrické vyhledávání bodu

*Vstup: Souvislá polygonová mapa  $n$  polygonů  $\{P_1, \dots, P_n\}$ , analyzovaný bod  $q$ .*

*Výstup:  $P_i, q \in P_i$ .*

Nad polygonovou mapou implementujete následující algoritmy pro geometrické vyhledávání:

- Ray Crossing Algorithm (varianta s posunem těžiště polygonu).
- Winding Number Algorithm.

Nalezený polygon obsahující zadaný bod  $q$  graficky zvýrazněte vhodným způsobem (např. vyplněním, šrafováním, blikáním). Grafické rozhraní vytvořte s využitím frameworku QT.

Pro generování nekonvexních polygonů můžete navrhnout vlastní algoritmus či použít existující geografická data (např. mapa evropských států).

Polygony budou načítány z textového souboru ve Vámi zvoleném formátu. Pro datovou reprezentaci jednotlivých polygonů použijte špagetový model.

### Hodnocení:

Krok	Hodnocení
Detekce polohy bodu rozlišující stavy uvnitř, vně na hranici polygonu.	10b
Ošetření singulárního případu u Winding Number Algorithm: bod leží na hraně polygonu.	+2b
Ošetření singulárního případu u obou algoritmů: bod je totožný s vrcholem jednoho či více polygonů.	+2b
Zvýraznění všech polygonů pro oba výše uvedené singulární případy.	+2b
Algoritmus pro automatické generování nekonvexních polygonů.	+5b
<b>Max celkem:</b>	<b>21b</b>

Obrázek 1.1: Zadání cíčení č. 1 [1]

## 1.1 Údaje o bonusových úlohách

Cílem úlohy je představit grafickou aplikaci na geometricko detekci polohy bodu vůči polygonu, která rozlišuje stavy jako uvnitř, vně, a na hranici. Bonusovou úlohou je myšlená širší funkcionalita aplikace. Do aplikace bylo zakomponováno ošetření singulárního případu u Winding Number Algorithm - bod ležící na hraně polygonu, dále pak ošetření singulárního případu u Winding Number Algorithm a Ray Crossing Algorithm - bod je totožný s vrcholem jednoho nebo více polygonů, a na nakonec grafické zvýraznění polygonů do kterých spadá bod  $Q$  včetně ošetření obou singulárních případů [1].

## Kapitola 2

# Popis problému

V rovině máme mapu tvořenou  $j$  polygony, které se stávají z  $i$  vrcholů a  $\sigma$  hran. Do této mapy chceme umístit bod  $Q$  a určit jeho polohu vůči polygonům v mapě. Při určování polohy se výpočet provádí pro každý polygon zvlášť a na základně tohoto určení se rozhodne o poloze bodu vůči polygonu/polygonům ve který se bod  $Q$  nachází.

- Bod  $Q$  neleží v polygonu  $p[j]$
- Bod  $Q$  leží v polygonu  $p[j]$
- Bod  $Q$  leží na hraně  $\sigma$  polygonu/polygonů
- Bod  $Q$  je totožný s vrcholem  $v[i]$  polygonu /polygonů

Polohu bodu  $Q$  v mapě určíme algoritmy Winding Number Algorithm a Ray Crossing Algorithm. Tyto algoritmy byly zvoleny z důvodu své relativní výpočetní jednoduchosti.

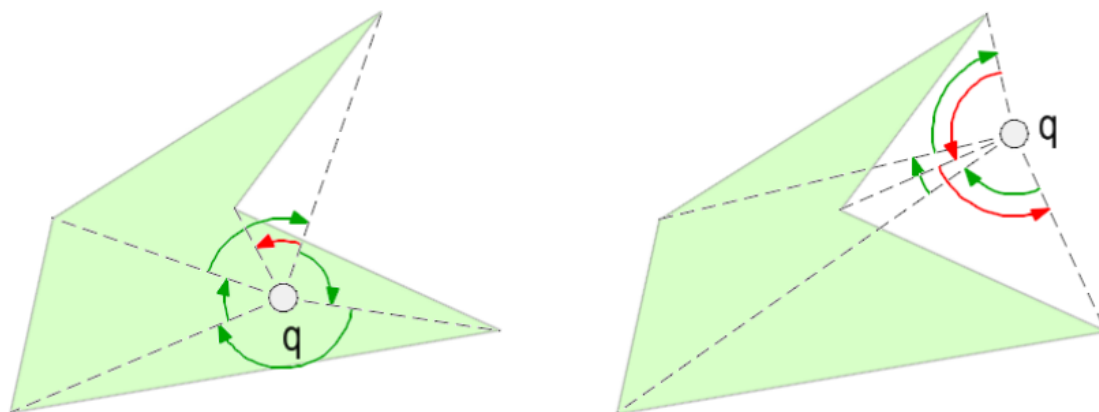
## Kapitola 3

# Popis použitých algoritmů

Pro určování polohy bodu  $Q$  v mapě tvořené polygony lze uplatnit mnoho algoritmů. Vždy je třeba zvolit vhodný algoritmus, který nebude příliš náročný na tvorbu a výpočet. Z tohoto důvodu byly zvoleny algoritmy Winding Number Algorithm a Ray Crossing Algorithm. Pro řešení úlohy by bylo možné využít také algoritmů Line Sweep Algorithm, Divide and Conquer nebo Brute Force Algorithm [2].

### 3.1 Winding Algorithm

Tato metoda spočívá ve sčítání úhlu, který svírají spojnice vrchol  $v[i]$  s určovaným bodem  $Q$ . Pokud se otáčíme ve směru hodinových ručiček od bodu  $v[i]$  k bodu  $v[i+1]$ , tak má úhel kladné znaménko, v opačném případě má znaménko záporné. Pokud výsledný úhel je roven  $2\pi$ , tak se bod nachází uvnitř polygonu  $Q \in p[j]$ , jinak polygonu  $Q \notin p[j]$ .



Obrázek 3.1: Princip Winding Algorithm [2]

Při práci s algoritmy vyvstávají různé problematické situace, je tedy nutné udělat jejich rozbor a ošetřit tyto situace v kódu. K problematickým situacím při řešení této úlohy může dojít,

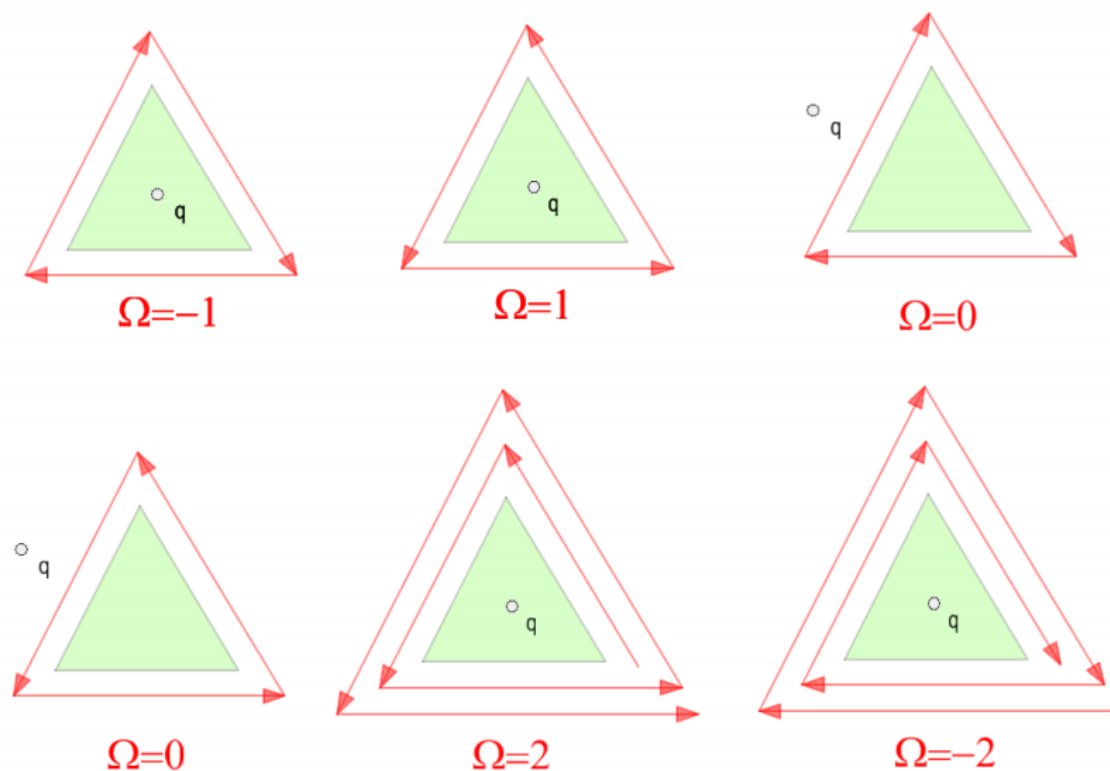
když bod leží na hraně polygonu nebo je totožný s některým vrcholem polygonu. Takovéto situace je potřeba ošetřit, aby i v takovém případě došlo ke správnému výběru polygonu.

### 3.1.1 Problematické situace u Winding Algorithm

V případě, že by bod  $Q$  ležel na některém z vrcholů polygonu, by algoritmus vyhodnotil, že bod leží mimo polygon, jelikož by součet úhlů vyšel menší než  $2\pi$ . Takovou situaci nastane když  $q[x_q, y_q] \approx p_i[x_i, y_i]$  a je potřeba v kódu ošetřit podmínkou  $|x_p - x_Q| < \epsilon \wedge |y_p - y_Q| < \epsilon$ . Pokud je tato podmínka splněna, tak výsledkem je, že se bod nachází uvnitř polygonu.

Pro případ, když bod  $Q$  leží na hraně  $|AB|$  polygonu, dojde u výpočtu, že bod leží vně polygonu, jelikož součet úhlů vyjde  $\pi$  nebo  $3\pi$ . Tato situace se ošetřuje podmínkou, kdy platí  $(|AB| - (|AQ| + |BQ|)) < \epsilon$ .

### 3.1.2 Implementace algoritmu



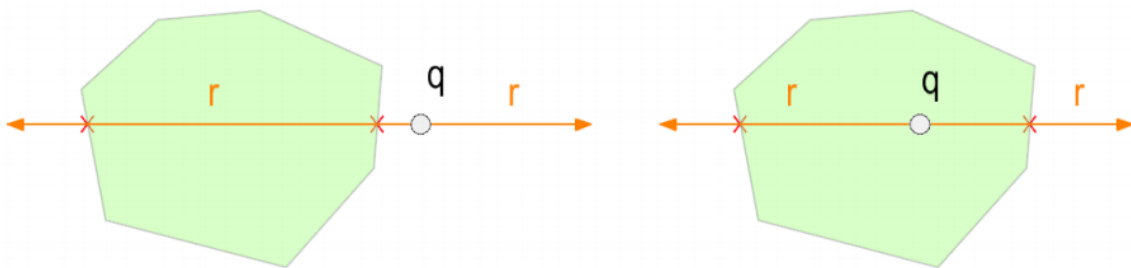
Obrázek 3.2: Princip Winding Algorithm [2]

1. Nastavení výchozího úhlu  $\omega$  rovno 0, volba tolerance  $\epsilon : \omega = 0, \epsilon = 1e - 10$
2. Určení orientace  $o_i$  bodu  $q$  ke straně  $p_i, p_{i+1}$

3. Určení úhlu  $\omega_i = p_i, q, p_{i+1}$
4. Volba podmínky - pokud pod vlevo:  $\omega = \omega + \omega_i$
5. V opačném případě:  $\omega = \omega - \omega_i$
6. Volba podmínky - pokud rozdíl:  $(\omega - 2\pi) < \epsilon$ , pak platí:  $q \in P$
7. V opačném případě:  $q \notin P$

## 3.2 Ray Crossing Algorithm

Metoda Ray Crossing Algorithm, česky volně přeloženo jako průsečík paprsku, vychází z určování počtu průsečíků hran polygonu a polopřímky vycházející z libovolného bodu. Když jako výchozí bod polopřímky nebo-li paprsku (ray) zvolíme bod  $Q$ , můžeme tak určit jeho polohu vůči polygonu. Počet průsečíků si označíme jako  $k$ . Pokud je  $k$  liché, tak bod leží uvnitř polygonu  $Q \in p[j]$ , jinak leží vně  $Q \notin p[j]$ .



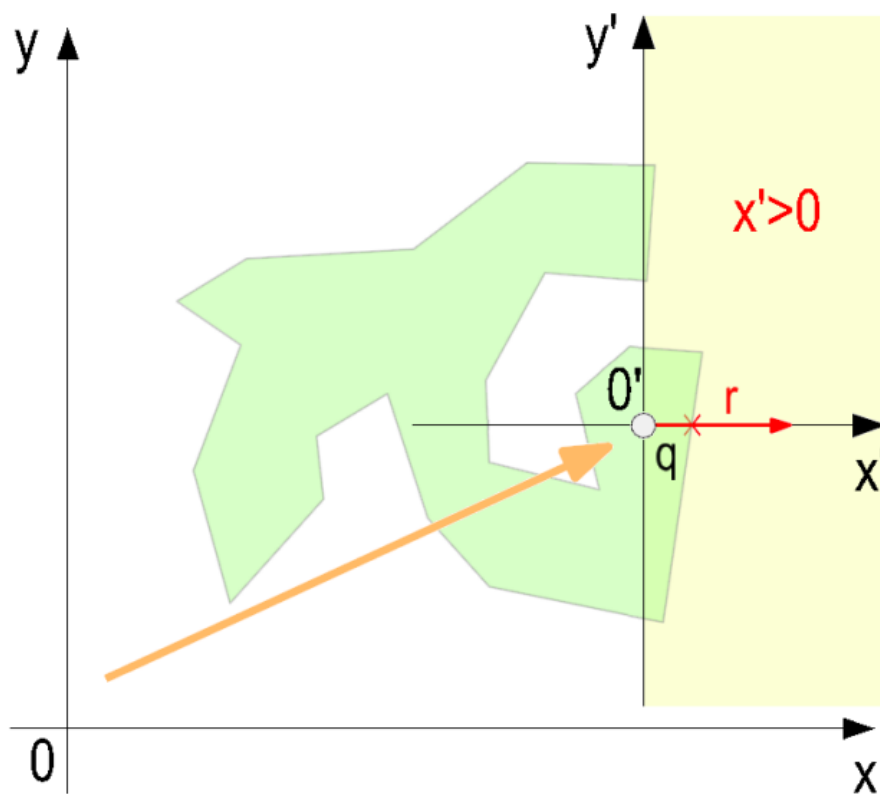
Obrázek 3.3: Princip Ray Crossing Algorithm [2]

### 3.2.1 Problematické situace u Ray Crossing Algorithm

Při této metodě může dojít takové k situaci, že některá hrana polygonu leží na polopřímce z bodu  $Q$ . V takovém případě dochází k tomu, že počet průsečíků polopřímky s hranami polygonu je sudý, i když je bod  $Q$  uvnitř polygonu. Z tohoto důvodu se zavádí redukovaný souřadnicový systém s počátkem v bodě  $Q$ , kde osa  $x'$  je rovnoběžná s osou  $x$  a osa  $y'$  je kolmá na  $x'$ . Řešení úlohy se poté omezí pouze na hrany, jejichž jeden vrchol leží pod osou  $x'$  a druhý nad osou  $x'$ .

U metody Ray Crossing Algorithm se problematické situace řeší principiálně stejně jako u Winding Number Algorithm, viz. 3.1.1.





Obrázek 3.4: Princip Ray Crossing Algorithm - redukované souřadnice[2]

### 3.2.2 Implementace metody

1. Nastavení počtu průsečíků rovno nule:  $inters = 0$
2. Redukce souřadnic x všech bodů polygonu vůči x-ové souřadnici bodu q:  $x'_i = x_i - x_q$
3. Redukce souřadnic y všech bodů polygonu vůči y-ové souřadnici bodu q:  $y'_i = y_i - y_q$
4. Volba podmínky:  $if(y'_i > 0) \wedge (y'_{i-1} \leq 0) \parallel (y'_{i-1} > 0) \wedge (y'_i \leq 0)$
5. Při splnění podmínky:  $x'_m = (x'_i y'_{i-1} - x'_{i-1} y'_i) / (y'_i - y'_{i-1})$
6. Pokud  $x'_m > 0$ , zvýšení počtu průsečíků o jeden:  $inters = inters + 1$
7. Určení zda počet průsečíků sudý či lichý:  $if(inters \% 2) = 0$ , pak:  $q \in P$  - počet průsečíků je sudý
8. V opačném případě:  $q \notin P$

## Kapitola 4

# Vstup a výstup aplikace

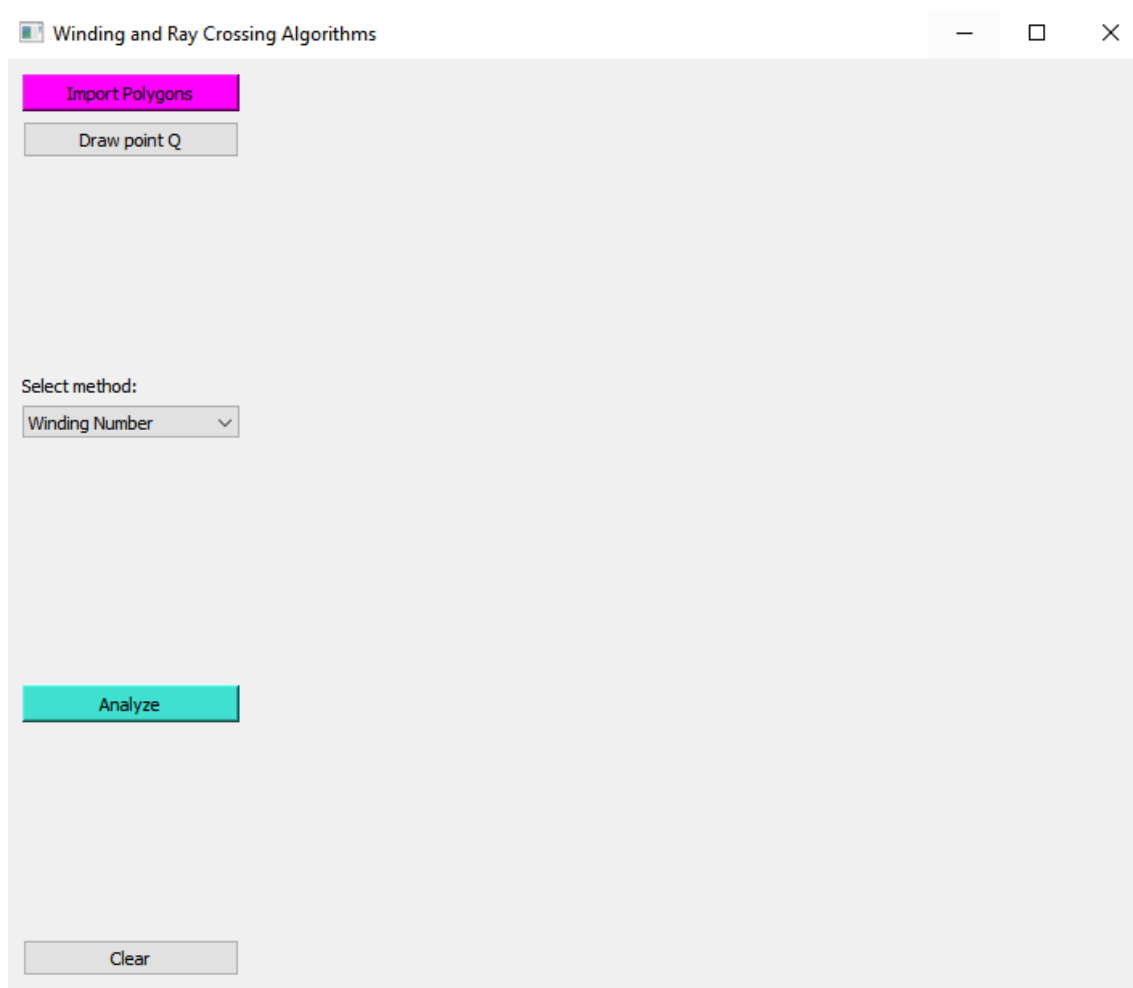
### 4.1 Vstup

- Vstupními daty pro tuto úlohu je textový soubor formátu .txt a bod  $Q$ . Textový soubor je potřeba do aplikace načíst pomocí tlačítka "Import Polygons". Soubor obsahuje body jednotlivých polygonů v řádcích ve formátu [číslo bodu, souřadnice X, souřadnice Y]. Každý polygon začíná bodem 1 a končí číslem  $n$ -tým bodem, pokud v souboru je načítán bod s číslem 1, dojde k ukončení předchozího polygonu a začíná nový polygon. Čísla bodů a souřadnice jsou v souboru odděleny mezerami.
- Bod  $Q$  do aplikace vloží uživatel kliknutím levým tlačítkem myši po aktivování přes tlačítko "Draw point Q".

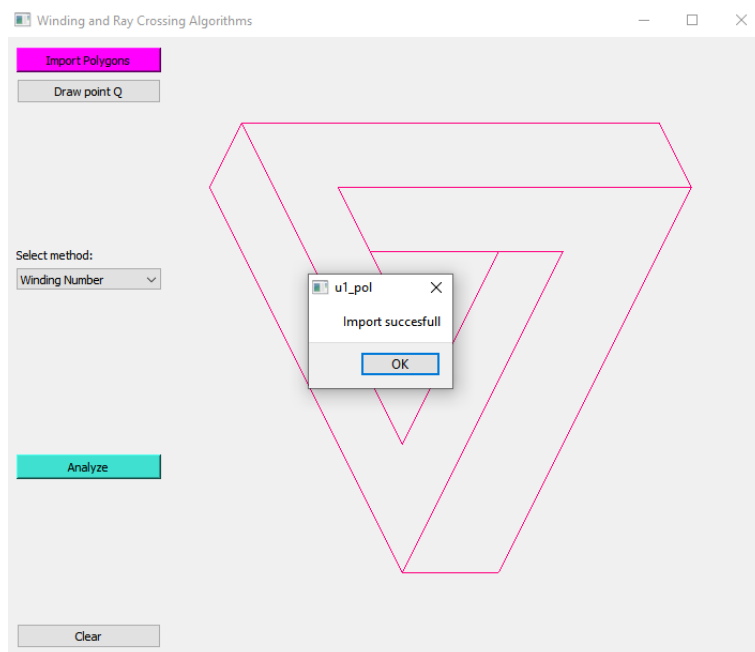
### 4.2 Výstup

- Výstupem úlohy je grafická aplikace, která po načtení souboru s polygony a volbě bodu  $Q$  určí polygon, ve kterém se bod nachází. Pro analýzu polohy bodů je možné zvolit ze dvou algoritmů Winding Number Algorithm a Ray Crossing Algorithm.
- Polygon, ve kterém se bod nachází, se po stisknutí tlačítka "Analyze" vybarví.
- Pokud je bod na hraně dvou polygonů nebo je totožný s vrcholem více polygonů, tak se vybarví všechny dotčené polygony.
- Pokud se bod nenachází v žádném polygonu, tak se žádný polygon nevybarví.
- Grafický výstup se pak stiskem tlačítka "Clear" smaže.

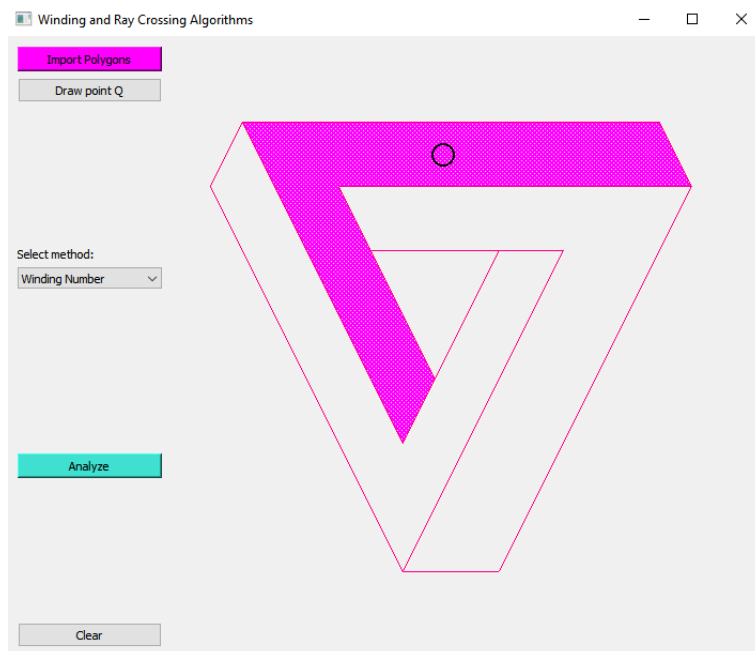
## 4.3 Prostředí



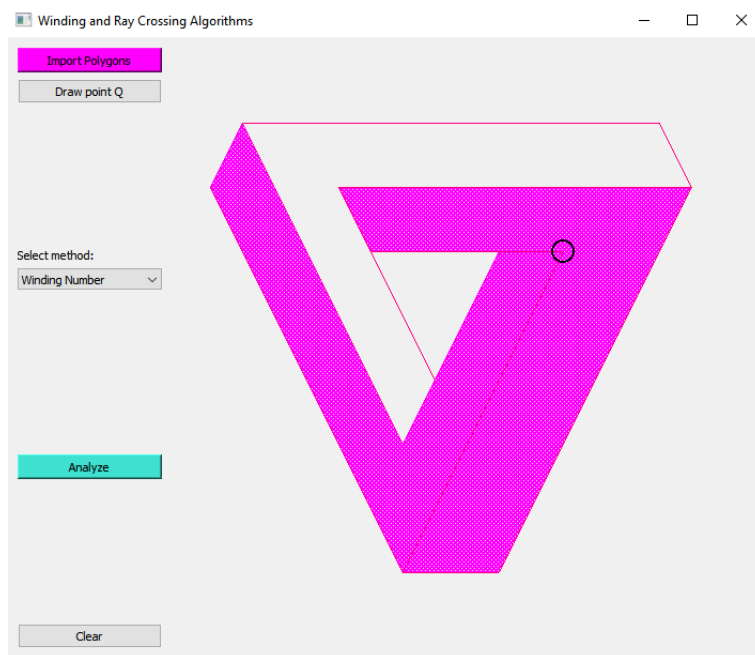
Obrázek 4.1: Prostředí po spoštění aplikace



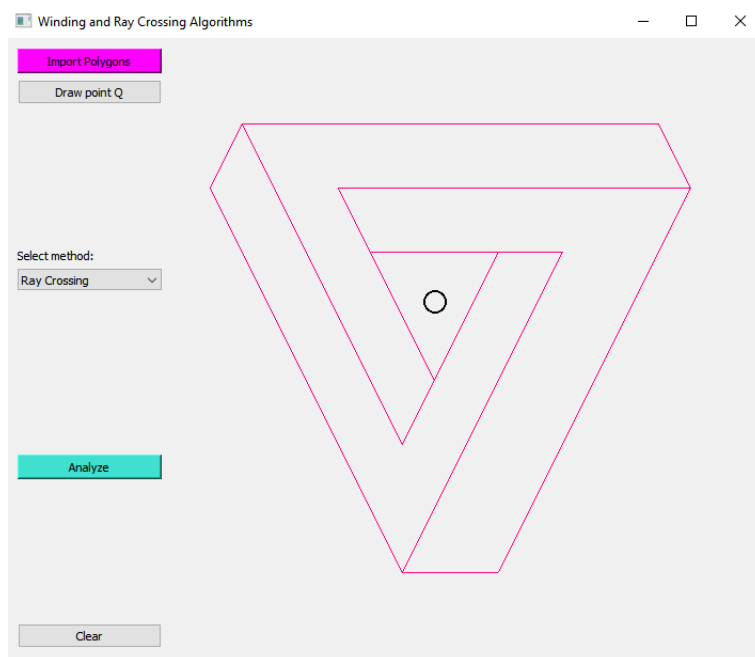
Obrázek 4.2: Hláška úspěšného importu



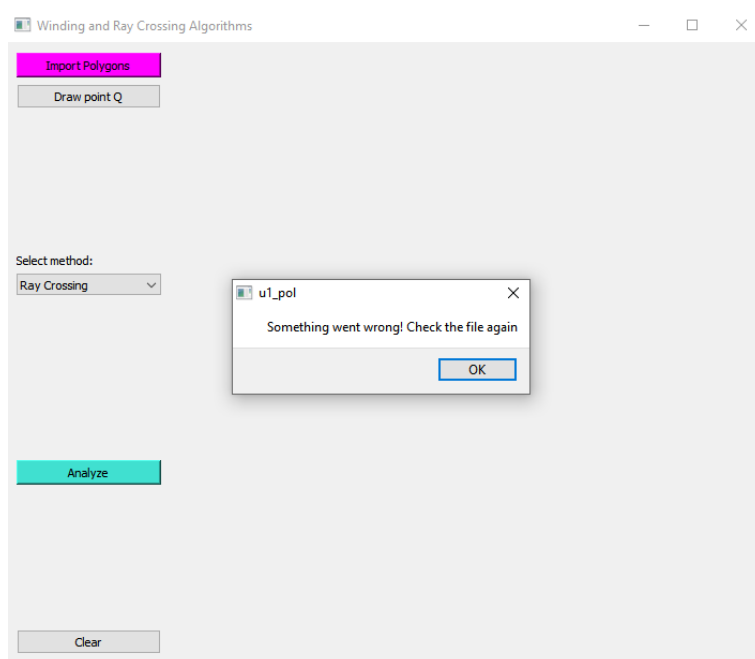
Obrázek 4.3: Vybarvení polygonu obsahujícího bod Q



Obrázek 4.4: Vybarvení dvou polygonů s dotykem



Obrázek 4.5: Analýza bodu mimo polygony



Obrázek 4.6: Hláška neúspěšného importu

## Kapitola 5

# Dokumentace tříd a metod

### 5.1 Algorithms

- *int getPositionWinding(QPointF q, QPolygonF pol)*  
Funkce, která určuje polohu bodu vůči polygonu metodou Winding Number Algorithm. Na vstupu funkce je určovaný bod q a polygon P. Výstupem je celé číslo a to 1(bod leží uvnitř polygonu, na hraně nebo na vrcholu), 0(bod leží vně polygonu) nebo -1(jiné).
- *int getPositionRay(QPointF q, QPolygonF pol)*  
Funkce, která určuje polohu bodu vůči polygonu metodou Ray Crossing Algorithm. Na vstupu funkce je určovaný bod q a polygon P. Výstupem je celé číslo a to 1(bod leží uvnitř polygonu, na hraně nebo na vrcholu) nebo 0(bod leží vně polygonu).
- *int getPointLinePosition(QPointF  $\mathcal{E}q$ , QPointF  $\mathcal{E}a$ , QPointF  $\mathcal{E}pol$ )*  
Tato funkce určuje pozici bodu vůči linii. Na vstupu funkce je určovaný bod a dva body přímky "a" a "b". Ze dvou bodů přímky můžeme určit determinant, jehož hodnotu porovnáváme se zvolenou minimální hodnotou "eps". Výstupem funkce je celé číslo, které nabývá hodnot 1(pod leží v levé polorovině), 0(bod leží v pravé polorovině) a -1(bod leží na hraně).
- *double get2LinesAngle(QPointF  $\mathcal{E}p1$ , QPointF  $\mathcal{E}p2$ , QPointF  $\mathcal{E}p3$ , QPointF  $\mathcal{E}p4$ )*  
Tato funkce počítá úhel mezi dvěma hranami. Na vstupu jsou 4 body, které určují dvě přímky. Výstupem je velikost úhlu ve stupních typu double.

### 5.2 Draw

- *void paintEvent()*  
Metoda, která vykresluje naimportované polygony a bod q. Dále vykreslí polygon, uvnitř kterého se nachází bod Q.
- *void mousePressEvent()*  
Po kliknutí myši uloží souřadnice bodu Q.

- *void setDrawPoint*  
Slouží k zapnutí vykreslování bodu  $Q$ . Po spuštění programu lze vytvářet bod  $q$  až po stisknutí tlačítka Draw Point. Po obětovném stisknutí je vytváření bodu  $q$  opět vypnuté.
- *void clearCanvas*  
Slouží k vyčištění grafického okna od všech polygonů i bodu  $Q$ .
- *bool importPolygons*  
Používá se k načtení textového souboru s body polygonů.

### 5.3 Widget

- *void on\_pushButton\_3\_clicked*  
Vyčistí okno aplikace. V aplikaci se jedná o tlačítko "Clear".
- *void on\_pushButton\_clicked*  
Slouží k tvorbě bodu  $q$ . V aplikaci se jedná o tlačítko "Draw Points".
- *void on\_pushButton\_2\_clicked*  
Spustí výpočet algoritmu pro určení polohy bodu. V aplikaci se jedná o tlačítko "Analyze".
- *void on\_importPolygons\_clicked*  
Při stisknutí tlačítka se otevře dialogové okno pro vyhledání souboru se souřadnicemi bodů polygonů. V aplikaci se jedná o tlačítko "Import Polygon".



# Kapitola 6

## Závěr

Do vytvořené aplikace lze načíst data polygonů, vytvořit bod  $Q$  a spočítat zda bod leží uvnitř polygonu. Aplikace vybarví Polygon, který obsahuje bod  $Q$ . V programu jsou vyřešeny i případy, kdy se bod nalézá na hraně nebo vrcholu jednoho nebo více polygonů.

### 6.1 Neřešené problémy a náměty

- Z časové tísně nebyla v úloze vyřešena tvorba nekonvexních polygonů.
- Data musí být předem upravena, jelikož program má předdefinovanou velikost vykreslovacího okna. V ideálním případě by bylo dobré kdyby se načtená data sama přizpůsobila velikosti okna.
- Praktické by též bylo rozšířit aplikaci o funkci jež převádí kartézské souřadnice do souřadnic obrazových.

# Literatura

- [1] T. Bayer. Geometrické vyhledávání bodu, cvičení č. 1 předmětu Algoritmy v digitální kartografii, 2016. <https://web.natur.cuni.cz>.
- [2] T. Bayer. Geometrické vyhledávání bodu, přednáška č. 2 předmětu Algoritmy v digitální kartografii, 2016. <https://web.natur.cuni.cz>.

## Příloha A

# Zdrojový kód aplikace

Zdrojový kód aplikace je dostupný z veřejného profilu *petrposkocil* na github.com

Jméno repozitáře: *ADK\_poskocil\_faber*

Podsložka: *u1\_pol*

[https://github.com/petrposkocil/ADK\\_poskocil\\_faber/U1/u1\\_pol](https://github.com/petrposkocil/ADK_poskocil_faber/U1/u1_pol)

### Obsažené soubory:

*1\_pol.pro* //Qt creator project file

*algorithms.h* //Header file

*draw.h* //Header file

*widgets.h* //Header file

*algorithms.cpp* //Source code file

*draw.cpp* //Source code file

*widgets.cpp* //Source code file

*widget.ui* //User interface file

## Příloha B

# Vstupní a výstupní data

Vstupní data jsou dostupná z veřejného profilu *petrposkocil* na github.com

Jméno repozitáře: *ADK\_poskocil\_faber*

Soubor: *imput\_data.txt*

[https://github.com/petrposkocil/ADK\\_poskocil\\_faber/U1/imput\\_data.txt](https://github.com/petrposkocil/ADK_poskocil_faber/U1/imput_data.txt)

Výstupní data jsou ve formě grafického znázornění výsledků algoritmů. Polygon do kterého padnou souřadnice bodu  $Q$  je vyzobrazen barevnou šrafou.

## Příloha C

# Aplikace - binární soubor

Aplikace je dostupná z veřejného profilu *petrposkocil* na github.com

Jméno repozitáře: *ADK\_poskocil\_faber*

Soubor: *u1\_pol.exe*

[https://github.com/petrposkocil/ADK\\_poskocil\\_faber/U1/u1\\_pol.exe](https://github.com/petrposkocil/ADK_poskocil_faber/U1/u1_pol.exe)